Transferring Image-based Edits for Multi-Channel Compositing

JAMES W. HENNESSEY, University College London
WILMOT LI, BRYAN RUSSELL, and ELI SHECHTMAN, Adobe Research
NILOY J. MITRA, University College London

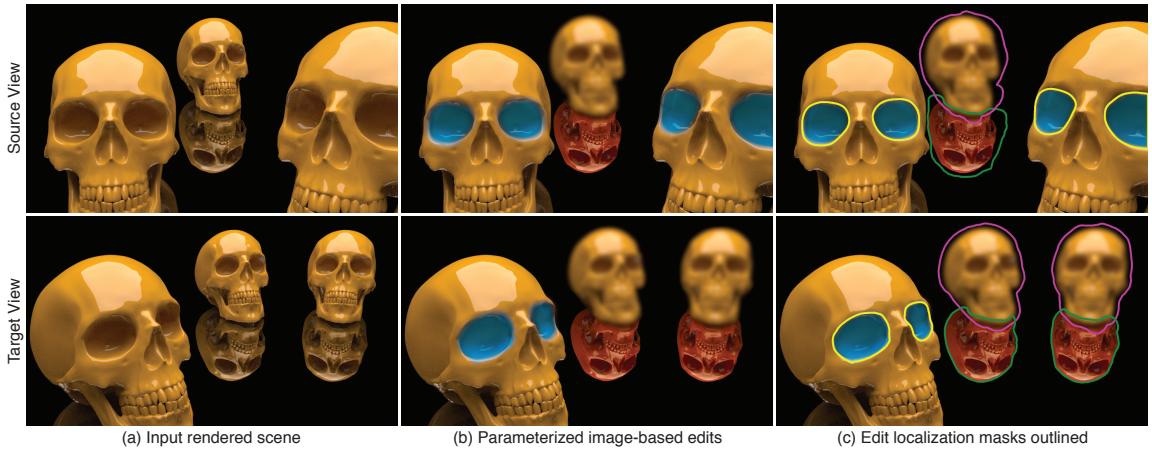


Fig. 1. (Top) Input source view rendered using a set of photometric render channels. (a) Composite of All Photometric channels. (b) The user applies 2D image-based edits to specified channels such as: blurring the background object to create depth of field effect (All Photometric channels); adjusting gamma, hue, and saturation to emphasise floor reflections (Reflection channel); Making the eye sockets of foreground skulls appear to glow blue by adjusting the hue, saturation, and lightness (Diffuse and Global Illum channels). (Bottom) Given a target view (a) with a different scene configuration (skulls are positioned in different 3D locations and orientations) (b) our method transfers the 2D image-based user edits automatically. The right column (c) shows the outlines of the corresponding localization masks for the two views. Multiple instances of the same object make this a challenging scene. For baseline comparisons, please see supplementary material.

A common way to generate high-quality product images is to start with a physically-based render of a 3D scene, apply image-based edits on individual render channels, and then composite the edited channels together (in some cases, on top of a background photograph). This workflow requires users to manually select the right render channels, prescribe channel-specific masks, and set appropriate edit parameters. Unfortunately, such edits cannot be easily reused for global variations of the original scene, such as a rigid-body transformation of the 3D objects or a modified viewpoint, which discourages iterative refinement of both global scene changes and image-based edits. We propose a method to automatically transfer such user edits across variations of object geometry, illumination, and viewpoint. This transfer problem is challenging since many edits may be visually plausible but non-physical, with a successful transfer dependent on an unknown set of scene attributes that may include both photometric and non-photometric features. To address this challenge, we present a transfer algorithm that extends the image analogies formulation to include an augmented set of photometric and non-photometric guidance channels and, more importantly, adaptively estimate weights for the various candidate channels in a way that matches the characteristics of each individual edit. We demonstrate our algorithm on a variety of complex edit-transfer scenarios for creating high-quality product images.

2019. Manuscript submitted to ACM

CCS Concepts: • **Computing methodologies → Image processing**; *Non-photorealistic rendering*;

ACM Reference format:
James W. Hennessey, Wilmot Li, Bryan Russell, Eli Shechtman, and Niloy J. Mitra. 2019. Transferring Image-based Edits for Multi-Channel Compositing. 1, 1, Article 1 (March 2019), 28 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Physically-based rendering algorithms have matured to the point where they are increasingly used to create photorealistic product images. For example, IKEA reports [FastCompany 2014] that 75% of their catalogue images are rendered rather than photographed. In addition to being more cost-effective than real photography, one key advantage of rendered content is that it provides artists with greater editing flexibility. While some edits can be easily achieved by changing 3D rendering parameters (e.g., changing the color or intensity of light sources), many other edits are not physically valid and are thus difficult to express in 3D (e.g., removing distracting reflections, emphasizing specific object contours). Artists typically make such non-physical edits in 2D by editing the individual render channels (e.g., Diffuse Lighting, Secular Reflections, Refractions, etc.) that together make up the final rendered result. The typical workflow is to mask out a specific element of the image, like a specific reflection or object contour, and then either mute or emphasize it by applying some parameterized adjustment (e.g., brightness, contrast, exposure, levels). Many rendered images are "retouched" in this manner to produce the final composited image. One such example is shown in Figure 1. There are several video tutorials demonstrating this workflow [3DArtist 2016; CGalter 2015].

While editing multi-channel renderings is a powerful approach, it also has some challenges. Most high-quality renderings include a large number of render channels (typically 4–15), which requires artists to flip through many channels to determine which one to edit. For many image editing experts who lack 3D rendering expertise, this task is especially difficult since they may have little intuition about which channels contribute to the image element they want to adjust. More importantly, once the artist has made edits on one rendered version of a scene, those edits cannot be re-used to create variations of the scene. For example, if a client or art director requests even small changes to the position or orientation of objects, lights or the camera, all the edits must be redone from scratch for the new scene configuration. Another common scenario is inserting or replacing objects in the scene. This unfortunate limitation adds significant inefficiencies to the authoring process and discourages iterative design space exploration for rendered product images.

In this work, we propose a novel compositing workflow that addresses these challenges. To retouch a rendered image, the user marks a region that requires an edit. Our system then automatically identifies suitable render channels to modify and, based on the selected channels, proposes a candidate mask (which the user can refine if necessary). The user can then make a number of parameterized adjustments - levels, exposure, gamma, blurring, hue, saturation, lightness - to modify the appearance of the masked region, and repeats this process until all the desired edits have been made. Given a modified version of the 3D scene, our system automatically transfers over all of the image-based edits, which allows users to quickly experiment with variations in viewpoint, object positions, object configurations (e.g., replacing an object), and lighting effects while preserving the image-based edits. For example, Figure 1 shows several edits to rendering with multiple instances of a skull being transferred to new scene configuration.

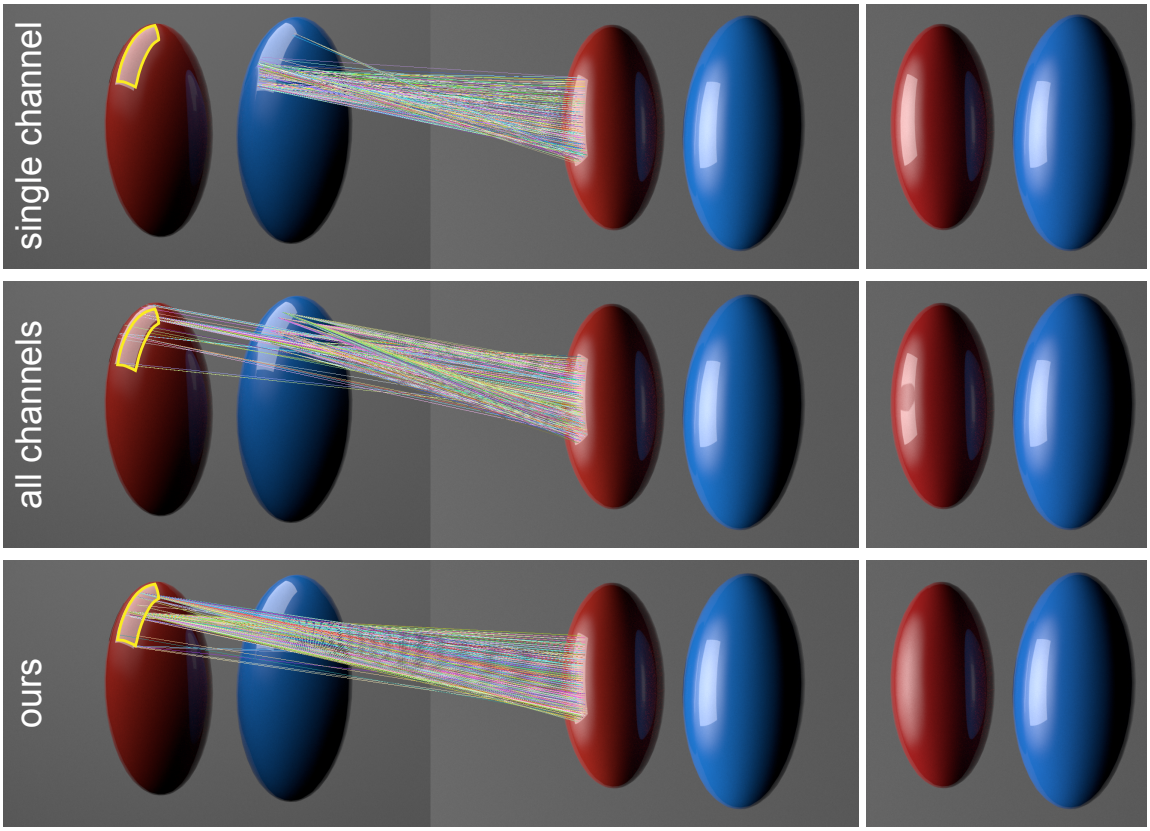


Fig. 2. Given an example edit for an input view (left) where the user masked out the reflection on the red object (outlined in yellow) to be removed, the challenge is to transfer the edit mask to a novel view (centre). Existing variants of image analogy can easily fail: (top) a single channel (reflection channel) is not sufficient as it wrongly establishes correspondence with the blue object; adding all the photometric channels (middle) with fixed weights is also not sufficient as the channels that are not relevant to the edit corrupt the correspondence, resulting in a bad mask transfer. (bottom) Our method, which adaptively estimates weights for the different channels to best explain the example edit, results in a successful edit transfer. The right column shows the resulting edit using the transferred mask.

The main technical challenge in supporting this workflow is how to perform the edit transfer. One approach is to formulate the task as an image analogies problem [Fišer et al. 2016; Hertzmann et al. 2001], where the input is the original rendered image (A), the edited image (A'), and an unedited rendering of the modified scene (B). The goal is to generate the analogous edited version of the modified scene (B'). Previous work demonstrates that providing the synthesis procedure with additional guidance channels (e.g., Photometric Render Channels, otherwise known as *light path expressions* [Heckbert 1990]) can be very effective. However, choosing the right guidance channels is not a trivial task. While the edited render channel is an obvious candidate, a single channel is often not sufficient to characterize the edit in a unique way. On the other hand, adding additional channels that are not correlated with the edit is problematic since they add noise and corrupt the signal of the correlated channels, hence can have a negative impact on the synthesized output. Figure 2 shows how such problems can arise even in a very simple editing scenario. In short, transferring image-based edits across different 3D scene configurations is a difficult task.

In our approach, we introduce a new image analogies formulation that automatically adapts the weights for a large set of candidate guiding render channels based on the characteristics of each edit. In particular, for each edit, we solve for a sparse set of render channels that best reconstruct the edit via L_1-regularized regression. This technique allows us to transfer edits that depend on a broad spectrum of different scene features (e.g., normals, depth, lighting effects, etc.). Furthermore, rather than synthesizing the appearance of edited image regions, we synthesize the edit masks and then solve for the appropriate adjustment parameters in the modified scene. This approach makes it convenient for users to refine the results by editing the transferred masks and parameters.

We evaluate our method on a range of challenging edit transfer scenarios under different scene variations involving object manipulation, illumination adjustment, and viewpoint changes. In most cases, the automatically transferred edits successfully reproduce the modifications to the original scene configuration and require no additional user refinement. In the few situations where the fully automatic transfers are not completely satisfactory, small tweaks to the synthesized edit masks or adjustment parameters are typically sufficient to achieve the desired result. We conducted a user study demonstrating significant time savings compared to manually transferring edits to different scene variations.

In summary, we present a novel editing workflow for multi-channel compositing; develop a smart selection tool for identifying relevant render passes and automatically creating corresponding local masks; and formulate an optimization for transferring local parametric edits in an adaptive Image Analogies framework.

2 RELATED WORK

2.1 3D Appearance Editing

There is a significant body of work on manipulating the appearance of rendered 3D objects. In particular, many of these methods help users adjust the output of physically-based rendering techniques via "artistic" controls, such as scribble based material appearance transfer [An et al. 2011], relighting a scene using a lighting paint brush [Pellacini et al. 2007], exploting image-space repetitions to transfer edits [Cheng et al. 2010], or using voice to interactively edit image edits [Cheng et al. 2014]. These are summarized in a recent survey by Schmidt et al. [2016]. While such controls are designed to facilitate the editing process, making specific adjustments to visual elements of a rendered scene is often still quite challenging given the complex interactions between light, materials, and geometry within most scenes. Moreover, many edits that artists want to make are either non-physical in nature (e.g., boosting and muting various highlights on an object) or much easier to specify in image space (e.g., emphasizing rim lighting at specific object contours). Finally, in some cases, the artist who creates the final, composited product image may simply have much more familiarity with 2D image editing tools than 3D software. As a result, we focus on image-based retouching workflows (as described in Section 1) rather than 3D appearance editing.

Another type of 3D appearance editing that is typically used for visual effects in computer-generated movies and animation is node-based compositing. For example, Nuke [TheFoundry 2017] is a popular commercial tool that supports this type of compositing. In such tools, masks are defined based on object or material ids, and a user-specified set of parameter adjustments are applied to the entire object or material based on these masks in each rendered frame. In contrast, our goal is to represent and transfer edits that are localized to specific parts of an object or material.

2.2 2D Edit Transfer

Previous work proposes a wide variety of techniques that facilitate image editing operations [Barnes et al. 2009; Cheng et al. 2010; Darabi et al. 2012; Diamanti et al. 2015; Hennessey and Mitra 2015; Levin et al. 2004; Reinhard et al. 2001;

Zheng et al. 2012]. The most relevant to our work are methods for transferring image edits across different images. Some approaches leverage inter-image correspondences to transfer edits to different viewpoints of the same scene or people [HaCohen et al. 2011, 2013; Hasinoff et al. 2010; Liu et al. 2011; Yücer et al. 2012, 2013]. However, even with access to perfect correspondences, such methods are not sufficient for our setting, since many retouching edits relate to lighting-dependent features (see Figure 9). An alternative approach is to use the editing history from the user interface [Berthouzoz et al. 2011; Grabler et al. 2009] or history inferred from the exemplar edit [Hu et al. 2013a] to transfer edits to new images. Our method is agnostic to the sequence of editing operations and only requires the final edited exemplar image. Our method builds on patch-based synthesis approaches [Barnes and Zhang 2017; Barnes et al. 2015; Hu et al. 2013b; Zhang et al. 2016] to transfer edits. More specifically, our contribution is a synthesis method adaptive to the user's edits.

We can also view image-based style transfer techniques as a form of edit transfer [Fišer et al. 2016; Gatys et al. 2016; Hertzmann et al. 2001]. However, such methods also have drawbacks. The deep learning-based neural style transfer of Gatys et al. [2016] can be difficult to control precisely; moreover, it may be hard to obtain sufficient training data to support our types of edits. Image analogies [Fišer et al. 2016; Hertzmann et al. 2001] provides a different formulation, but determining the appropriate guidance channels to successfully transfer edits is non-trivial, as noted already in Section 1 and in Figure 2. In addition, all of the aforementioned image-based techniques aim to transfer or synthesize the edit itself in the target image. In contrast, our goal is to transfer spatially localized parameterized edits that can be further refined by the artist.

2.3 Parametric Edit Transfers

Finally, some previous work has proposed techniques for transferring parameterized edits in the domains of 3D modeling and 2D vector graphics [Bernstein and Li 2015; Guerrero et al. 2016, 2014; Xing et al. 2014]. These methods demonstrate the utility of parameterized edit transfer for various content creation tasks. In our work, we present strategies for supporting a related type of edit transfer in the context of image-based edits to rendered content.

3 SYSTEM OVERVIEW

We illustrate our overall system in Figure 3. The input to our system is a 3D scene configuration that includes one or more objects at the desired positions and orientations, materials for those objects, a lighting setup, and a camera viewpoint. Such configurations can be created with most 3D modeling and rendering software (e.g., Maya, VRay). The user may also specify a background photograph in which to composite the rendered scene. Given this input, we provide an interactive editing tool that helps users specify and transfer parameteric image-based edits from the initial configuration of the input scene (*source view*), to a modified configuration (*target view*) that may involve a different viewpoint, lighting, object arrangement, or in some cases, new objects with similar geometry. We represent image-based edits as a 2D region mask that identifies the relevant part of the image to modify and a parametric adjustment that is applied within the mask. As mentioned earlier, each edit is applied to one or more specific render channels.

The main technical contribution is in our synthesis-based approach for transferring image-based edits from the source to the target view. Specifically, we introduce an adaptive version of Image Analogies that automatically determines how to weight various candidate guidance channels in order to transfer each edit. We also present an interface that helps users select and modify the appropriate render channels to specify the image-based edits in the source view. We now provide details for these two aspects.

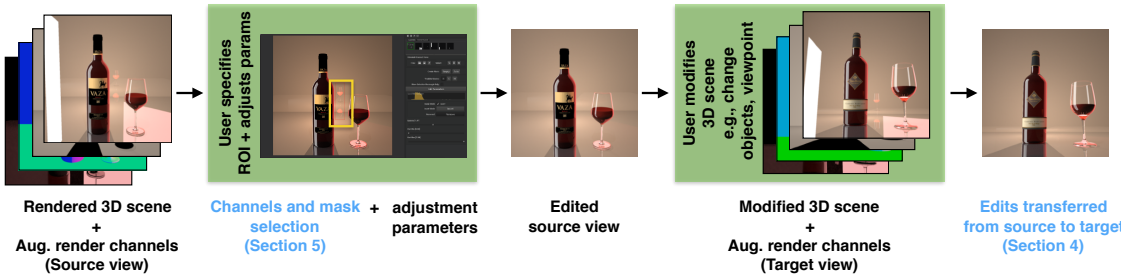


Fig. 3. System overview. Starting from an input source view of a rendered 3D scene, along with corresponding augmented render channels, the user may make a number of 2D edits. To make an edit, the user first outlines a region of interest (ROI). Our method then automatically determines a region mask and a selection of one or more relevant photometric render channels for the edit. The user then makes a parametric adjustment within the region mask to the selected channels to obtain an edited source view. In this example, the user removes the wine glass reflection and adjusts highlights on the labels. The user may then modify the 3D scene by replacing the 3D objects or changing the viewpoint to yield a target view. Our system automatically transfers the user edits from the source view to the target view. Text on green background denote the user interaction and blue text the computational aspects of our method.

4 TRANSFERRING PARAMETERIZED EDITS

Given a set of image-based edits in the source view, we transfer the edits to the target view in two stages. First, we transfer the 2D region mask to the target view using a new adaptive version of Image Analogies [Hertzmann et al. 2001]. Next, given the transferred region mask, we update the edit adjustment parameters for the target view. Before describing the details of mask and adjustment parameter transfer, we first introduce our set of augmented render channels that supports both of these steps.

4.1 Augmented Render Channels

The Image Analogies method is based on a repeated computation of a dense correspondence field (or nearest neighbor field) using guiding channels from the target to the source view. A critical challenge then is finding the right guiding channels resulting in a correspondence field that would be appropriate for the task of transferring edit masks and adjustment parameters. Note that the desired correspondence field may not simply be the rigid-body transformation of the 3D scene objects or a dense 3D correspondence field between two different 3D shapes. Many common edits, such as adjusting specular highlights or adding a halo around an object, may depend on one or more photometric and non-photometric factors, such as specularity, direction to light source, and the view-dependent silhouette of the object. Thus, our approach leverages a diverse set of rendered guidance channels derived from the 3D scene to help determine the correspondence between views.

Given a 3D asset with positioned lights or environment maps, we can output a full global illumination render of the 3D asset using a renderer such as VRay or Mitsuba [Jakob 2010]. Moreover, we can render a set of photometric channels, also called *light path expressions* [Heckbert 1990], that separate the different global illumination effects at each pixel. The different lighting effects can be diffuse or specular, and together sum to the full global illumination render of the 3D asset.

In addition to the standard set of photometric channels, we also render a set of complementary channels. Such channels are useful for finding edit-dependent dense correspondences. For example, the bottle rim lighting example in Figure 7 relies on the distance to the silhouette of the object. We render a number of channels relating to the 2D layout

and 3D geometry of the object, such as surface normals and distance transform to the object silhouette. Moreover, we found that including log-channels $\log(A_i + \epsilon)$, where A_i is a photometric render channel, boosts weak signals and improves transfer results. We used $\epsilon = 0.001$ for our experiments. Note that for augmented render channels with multiple dimensions at each pixel, we separate each dimension into its own augmented render channel. We normalize the Lab color channels into the range $[0, 1]$.

The success of our method does not rely on a specific set of render channels. The technique only requires a diverse superset of channels that are consistent between renderings. We demonstrate results using the VRay and Mitsuba renderers, which generate different sets of augmented render channels. Appendix B lists the specific set of channels for each renderer along with the render times.

4.2 Mask Synthesis via Adaptive Image Analogies

Given a set of augmented render channels $A = \{A_i\}$ for the rendered scene in the source view, a user edit e_A in the source view, and augmented render channels $B = \{B_i\}$ for the rendered scene in the target view, our goal is to infer the user edit e_B for the target view. We parameterize a user edit in the source view as $e_A = (X_A, A', \theta_A)$, where X_A are indices into the augmented render channels A indicating which photometric channels were selected by the user for the edit, A' is a real-valued user mask, and θ_A are parameters for the adjustment within the mask A' (see Section 5 for how edits e_A are specified using our interface). Similarly, we have $e_B = (X_B, B', \theta_B)$ for the target view. We assume that the selected photometric channels for the target view are the same as the source view, so we set $X_B \leftarrow X_A$. In this section we describe how to synthesize the user mask B' in the target view.

We formulate the mask synthesis task as one of finding an image analogy where $A : A' :: B : B'$ [Hertzmann et al. 2001]. While one could explicitly reason about the 3D scene via techniques for inverse rendering [An et al. 2011; Marschner 1998; Pellacini et al. 2007; Ramamoorthi and Hanrahan 2001; Schmidt et al. 2016] to recover the unknown mask, we argue that formulating the mask transfer task via image analogies is more flexible as it allows transfer of visually plausible but non-physically valid user edits.

The Image Analogies formulation proposed by Hertzmann et al. [2001] is a multi-scale iterative optimization algorithm. At each scale every iteration starts by computing a dense correspondence field given a previously computed B'. For every target patch around pixel q a best-matching source patch p is found that minimizes the following energy:

$$E_q(p) = ||A'(p) - B'(q)||^2 + \mu||A(p) - B(q)||^2, \tag{1}$$

where μ is a tunable scalar hyperparameter. Note that for the first iteration only the second term is used so that an initial B' mask can be synthesized. Given the dense correspondences, B' is updated by averaging the mask values for all overlapping best-matched patches for every pixel q. The overall energy is decreased after a few iterations and the result is upsampled to a finer scale until a solution (transferred mask and final correspondence field) at the finest scale is achieved. In [Hertzmann et al. 2001] the inputs are RGB images or steerable filter responses. More recently, Fišer et al. [2016] introduced StyLit, which uses photometric render channels as inputs to Image Analogies for illumination-guided stylization of 3D renderings. We build on the StyLit formulation for our task.

Transferring user-edit masks presents different challenges than the 3D rendering stylization transfer demonstrated in StyLit. As we will demonstrate in Section 6, simply applying the StyLit Image Analogies formulation produces a transferred edit mask with significant artifacts. We identify two reasons for this failure: (i) the information required for a particular edit transfer might not be present in the standard photometric render channels; and (ii) StyLit treats each

photometric render channel equally in the image analogies formulation. For example, to adjust a specular highlight, the system needs knowledge of not only the specular component, but also the direction to the light source. Moreover, not all photometric render channels are relevant to transfer the edit.

To address these issues, we leverage our augmented render channels to add non-photometric information that can aid in the transfer. To make use of the additional channels, we extend the standard image analogies formulation to one that adapts the weights of the different augmented render channels to a given user edit e_A:

$$E_q^{(e_A)}(p) = ||A'(p) - B'(q)||^2 + \mu \sum_i w_i^{(e_A)} ||A_i(p) - B_i(q)||^2, \tag{2}$$

where $\{w_i^{(e_A)}\}$ are given scalar weights for the augmented render channels dependent on user edit e_A.

4.3 Finding Edit-Dependent Weights

The adaptive edit-dependent image analogies energy in Equation (2) requires knowledge of a set of edit-dependent weights $\{w_i^{(e_A)}\}$, which guides the synthesis algorithm to know which augmented render channels are important for synthesis. We seek to automatically infer the edit-dependent weights given the user edit. This is challenging as we do not know a priori what type of edit the user is making, e.g., adjusting specular highlight or adding silhouette halo, or which channels are important for the edit.

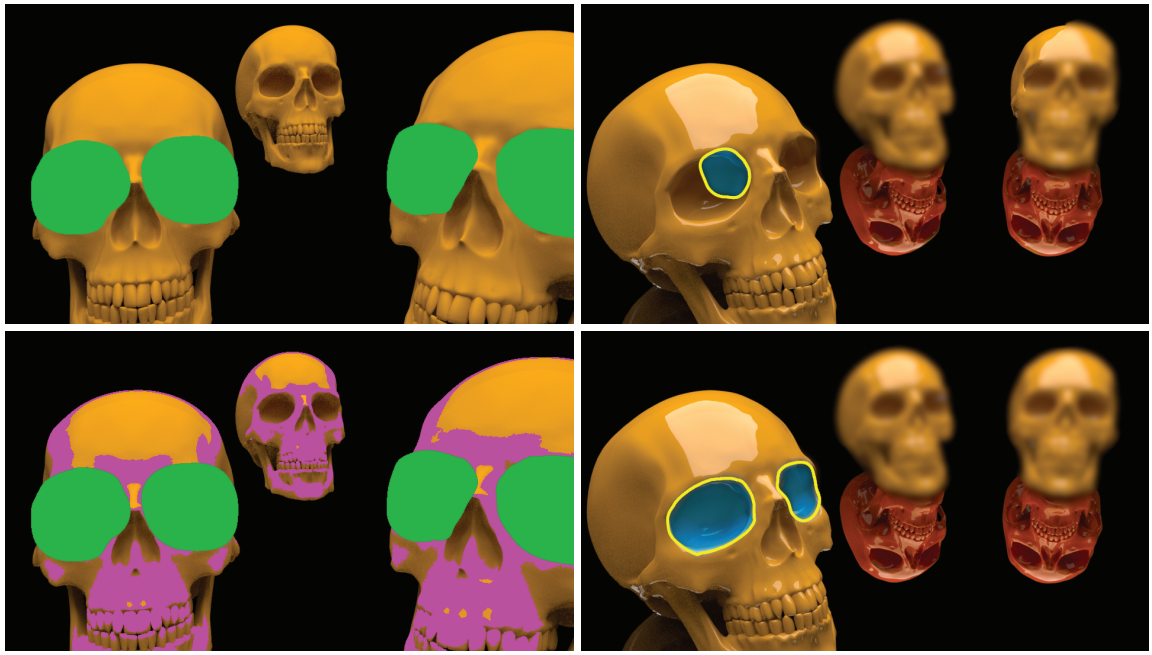
Since the desired weights are dependent on the user edit, and we do not have training examples with synthesized masks B' in the target view, we make the assumption that rendered channels important to synthesize B' in the target view are the same as the ones important to synthesize images of the user edit in the source view. As B' is related to the user edit, we find that this is a reasonable assumption that holds in practice and demonstrated in our final results. Moreover, we assume that not all channels are important and there can be some redundancy due to having an overcomplete superset of channels, meaning a sparse subset of all the channels will be sufficient to successfully transfer edits.

We formulate our edit-dependent weight recovery problem as an L_1-regularized regression to synthesize the user-edited source view. Let I_A be the image of the source rendered scene and $I_{A'}$ the image of the edited source rendered scene. We define the source edit-difference image at pixel location p as $\Delta_A(p) = I_{A'}(p) - I_A(p)$. We seek to find the weights $\boldsymbol{w}^{(e_A)} = [w_1^{(e_A)}, \ldots, w_N^{(e_A)}]^T$ that reconstructs the source edit-difference image from the augmented render channels $\{A_i\}$ for a set of sampled pixel location S,

$$\boldsymbol{w}^{(e_A)} \leftarrow \underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_{p \in S} \left(\sum_i A_i(p) w_i - \Delta_A(p) \right)^2 + \lambda ||\boldsymbol{w}||_1. \tag{3}$$

We use an L_1 sparsity prior over the edit-dependent weights, weighted by hyperparameter λ to select augmented render channels important for reconstructing the edit-difference image. Note that we use the Lab lightness channel for Δ_A instead of all color channels as we found that reconstructing the lightness channel provides better correspondences when there is a significant change in color in the target view.

4.3.1 Sampling. Since there are often many fewer non-zero pixels in the edit mask A' (dubbed *inside mask pixels*; we dub the complement set as *outside mask pixels*), we do not regress over the entire edit-difference image Δ_A. Instead, we balance the number of inside and outside mask pixels by including only the hardest outside mask pixels. The union of the inside and hardest outside mask pixels form the set of pixels S. Formally, let k be the number of inside mask



(a) Sampling Strategies (b) Transferred edits w/ localization mask outlined

Fig. 4. Using the source and target views from Figure 1 we show: (a) sampling strategies overlaid on the source view (showing user-selected photometric channels), and (b), the target view with resulting transferred edits. We only show the sampling and outline of the transferred mask for a single edit (eye sockets). (Top) Naively sampling only the masked region and immediate surrounding pixels (green) results in channels with non-zero weights (Log Specular, Log Reflection, Light Direction and Half Angle) that fail to uniquely describe the users edit and to transfer the masks. (Bottom) Our sampling additionally samples hard negatives (magenta), resulting in selection of render channels that characterizes the edit (Log Lighting, Shadow, Log Shadow and z-depth), and successfully transferring the edits.

pixels. We find the k outside mask pixels that are closest to the mean vector of the user-selected photometric channels $\{A_i\}_{i \in X_A}$ for the inside mask pixels. We also found that including pixels around the edge of the inside mask pixels found via dilation improves results. We used a unit dilation kernel of 3×3 pixels. This sampling scheme helps find weights $\boldsymbol{w}^{(e_A)}$ that differentiate between regions with similar features for inside and outside mask pixels. Figure 4 demonstrates the ability of this sampling strategy to find features unique to the given edit.

4.4 Adjustment Parameter Transfer

Our goal is, given the synthesized user mask B', to recover the user adjustment parameters θ_B for the user mask. To aid in the recovery, we first seek to synthesize the image $\tilde{I}_{B'}$, which is an estimate of the edited target rendered scene $I_{B'}$. We can then estimate the adjustment parameters θ_B by optimizing over the adjustment that best matches the estimate image $\tilde{I}_{B'}$.

Fig. 5. Adjustment parameter transfer. (Left) Baseline where we simply copy the user-provided adjustment parameter from the source view. (Right) Our approach for adjustment parameter transfer. Notice that simply copying the parameter results in a brighter reflection, whereas our approach more closely matches the edited source view (Figure 7).

To synthesize the estimate image $\tilde{I}_{B'}$, we leverage the learned edit-dependent weights $\boldsymbol{w}^{(e_A)}$ to reconstruct the edit-difference image Δ_B from the augmented render channels $\{B_i\}$,

$$\Delta_B(p) = \begin{cases} \sum_i w_i^{(e_A)} B_i(p). & \text{if } p \in \{p | B'(p) \neq 0\} \\ 0. & \text{otherwise} \end{cases} \tag{4}$$

Here, we only synthesize within non-zero pixels in the synthesized mask B'. Given the image of the target rendered scene I_B, we obtain the estimate image $\tilde{I}_{B'}(p) = I_B(p) + \Delta_B(p)$ at pixel location p.

Given the estimated image of the edited target rendered scene $\tilde{I}_{B'}$, we can recover the adjustment parameters θ_B by minimizing the following energy,

$$\theta_B \leftarrow \underset{\theta}{\text{argmin}} \sum_{p \in \{p | B'(p) \neq 0\}} ||\tilde{I}_{B'}(p) - I_{B'}(p; \theta)||^2, \tag{5}$$

where the image of the edited target rendered scene is given by

$$I_{B'}(p; \theta) = I_B(p) + \sum_{i \in X_B} B'(p) \left(f_\theta \left(B_i, p \right) - B_i(p) \right), \tag{6}$$

where f_θ is a parameterized image adjustment function. We optimize the above objective via grid search over the parameter space θ_B. We provide details of these f_θ functions and for the grid search in Section 5.2 and Appendix C when we introduce our editing tool.

4.5 Implementation Details

User-specified masks can often be coarsely specified if a masked region in the user-selected photometric channels $\{A_i\}_{i \in X_A}$ is surrounded by black pixels. This is due to many parameter adjustments having no effect in these black regions. Including all of these masked pixels can lead to over-sampling pixels p where the edit-difference $\Delta_A(p)$ is zero. This can make Equation 3 ineffective at choosing relevant features. So as a pre-processing step we removed pixel locations p from the mask $A'(p)$ where $\sum_{i \in X_A} f_{\theta_A}(A_i, p)$ is less than 10^{-3}. In an additional pre-processing step for gaussian-blur edits we set $A_i(p) = f_{\theta_A}(A_i, p)$ as the blur operation has a spatial extent not captured in Equation 3 as it

does not take into account neighbouring pixels. This pre-processing allows for edits with a spatial extent to work in our formulation.

We used a CPU C++ implementation multi-scale guided synthesis algorithm [Fišer et al. 2016]. Similar to their method, we used a fixed patch size of 5×5 pixels and pyramid down-sampling ratio of 2. We ran synthesis up to 6 levels in the pyramid and used fewer levels if the down-sampled user mask comprised less than 30 pixels in a given level. We set the hyperparameter μ for the adaptive image analogies Energy (2) to $\mu = 3$ for the first level and $\mu = \frac{1}{3}$ for the last level, and linearly interpolated the intermediary levels. Intuitively, the hyperparameter setting μ at the different levels allow for more guidance over the features at the beginning, and later to previous level's mask B'.

Additionally, after each level in the adaptive image analogies pyramid, we discarded correspondences that went to pixel locations in the target view where all of the selected photometric channels were less than 10^{-3}. At a given level of the pyramid it may not matter masking a region that is nearly black. However, a problem arises when the mask propagates to later levels of the pyramid where it should not be masked but due to the decreasing μ parameter the correspondence does not update, leading to spurious artifacts. In the special case of an edit e_A with a spatial extent (e.g. blur), we apply the edit at the pixel location using the θ_A parameters before testing for small values.

Similar to StyLit, we initialized B' by randomly assigning from A'. Additionally, on the first iteration we applied no weighting to the $||A'(p) - B'(q)||^2$ term. Fišer et al. [2016] introduce a new way to compute a correspondence field from the target to the source view, that avoids "washout" and obvious repetition artifacts. Their solution involved multiple source-to-target search iterations that significantly slowed down the computation. Since these artifacts are less relevant for textureless masks as they are for RGB images, we use the regular target-to-source search [Hertzmann et al. 2001; Wexler et al. 2007] using PatchMatch [Barnes et al. 2009].

The hardest regions to find correct correspondences are mask boundaries due to the averaging of conflicting features in the image pyramid. In cases of underestimating the boundary location, as a post-processing step, we compute the mean of the selected render channels in the output mask. For for all mask boundary pixels, we allow the mask to grow if the neighbouring pixel in the selected render channels was within 0.1 distance to the mean, up to a maximum of 5 pixels. To optimize the L1-regularized regression Energy (3), we used the publicly available POGS solver[1]. For scenes rendered using VRay we set hyperparameter $\lambda = 10000 + 300 \cdot N_L$, where N_L is the number of lights in the 3D scene as additional light sources introduce additional channels requiring more regularization. For scene rendered with Mitsuba we set $\lambda = 20000$ as the number of channels is fixed.

5 INTERFACE

Our interface allows users to quickly select render channels to edit, generate masks, and set adjustment parameters. The user starts by loading a stack of photometric render channels into our interface (see Figure 3 and supplementary video). By default, the users are only shown the composited image, but can switch to other channels as desired. For each edit, they specify a rough region of interest on the composited image, and our method automatically selects a subset of channels (named X_A in Section 4.2). The user can verify the correct channels were chosen via hot keys and use the auto-mask feature to create an edit mask (named e_A Section 4.2). The user then edits the selected channels inside the masked region by adjusting some of the supported adjustment parameters (named θ_A Section 4.2). The user can perform multiple edits on the same example scene, and transfer them to other comparable scene variations.

[1] http://foges.github.io/pogs

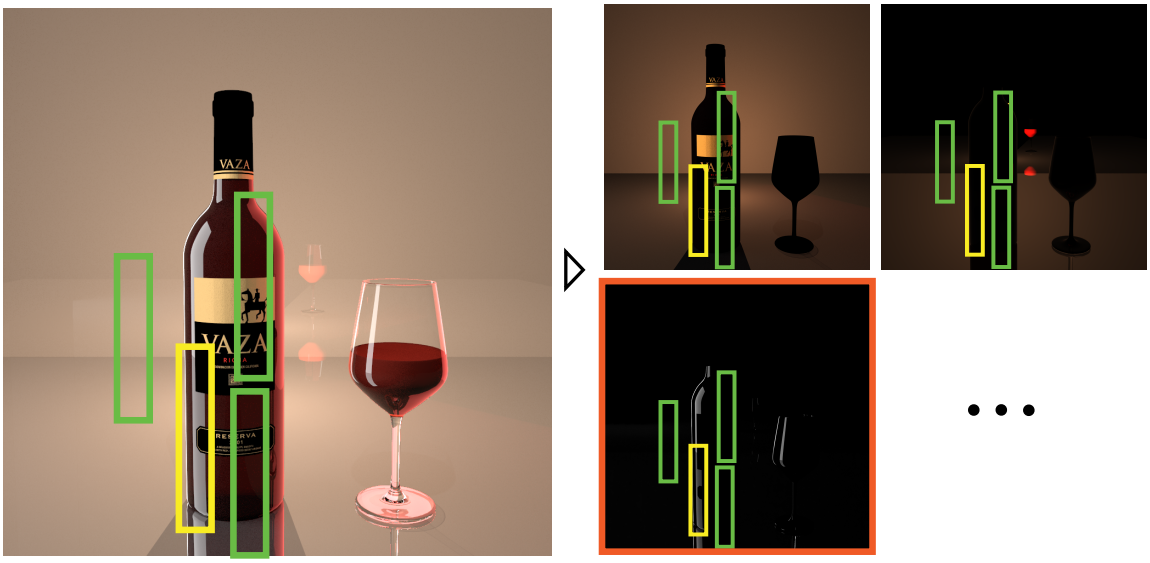


Fig. 6. (Left) The user marks a region of interest (ROI) $P_\star$ (shown as the yellow polygon) on the input composited image. (Right) By comparing patch statistics against neighboring regions P_j (shown as green polygons), our method automatically chooses which render channel(s) maximizes the uniqueness of ROI $P_\star$. In this example the reflection channel (highlighted in orange) was chosen.

5.1 Render Channel Selection For Editing

In order to select a subset of render channels, the user simply specifies a coarse region of interest (ROI) using either a rectangular marquee or polygon selection tool directly on the the final composited image. Our selection method then identifies the relevant channels based on the assumption that the user is interested in only those layers that make the selected region *unique* with respect to the neighboring regions. In the following, we first describe how to sample neighboring regions, formulate the selection problem given a choice of such neighboring regions, and finally create the edit mask (see Figure 6).

5.1.1 Sampling neighboring regions. Let $P_\star$ denote the set of all pixel locations in the user-selected ROI. Given $P_\star$, we first sample other regions at random by displacing the ROI by random translations with magnitude in the range $[\delta, 2\delta]$ with δ denoting the diameter of ROI $P_\star$ bounding circle. From the random samples we remove overlapping selections and those intersecting the ROI to generate m candidate (neighboring) patches $\{P_1, \ldots, P_m\}$, where P_j denotes a set of pixel locations in the jth neighboring region.

5.1.2 Selecting among the render channels. Among the channels $\{A_i\}_{i=1}^N$, we seek to identify the ones that are distinct within the ROI with respect to the spatially neighboring regions. Similar to approaches for bottom-up saliency [Itti and Koch 2001], we measure distinctness for a channel A_i by computing a difference between the statistics within ROI $P_\star$ and all neighboring regions P_j.

Let $\mu_i(P) = \mathbf{E}_{p\sim P}[A_i(p)]$ be the mean value within the render channel A_i for pixel locations P, and $\sigma_i^2(P) = \mathbf{E}_{p\sim P}[(A_i(p) - \mu_i(P))^2]$ the variance, stored as a vector of statistics $(\mu_i(P), \sigma_i(P))$. We define the difference between the statistics within the ROI $P_\star$ and neighboring region P_j for channel A_i using $d_{j,i} = \|(\mu_i(P_j), \sigma_i(P_j)) - (\mu_i(P_\star), \sigma_i(P_\star))\|$

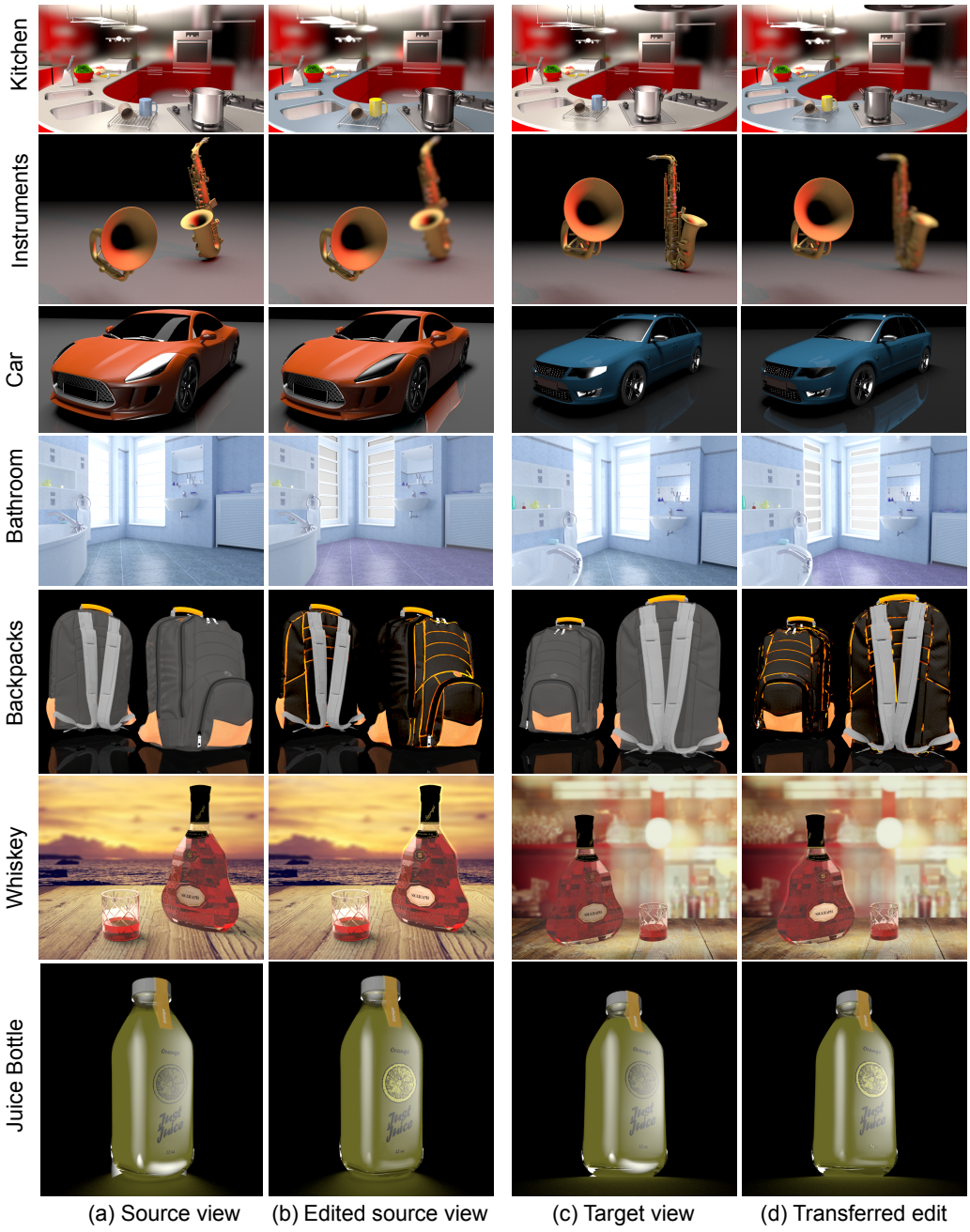


Fig. 7. Our results. To best view the transferred edits, please see the electronic paper version, the supplemental video, and the suppl. PDF.

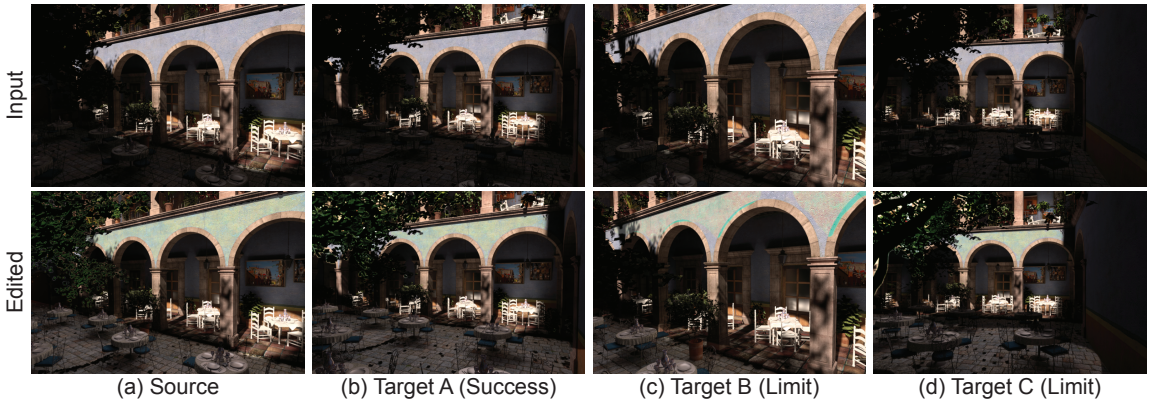


Fig. 8. Testing Limits: (a) Given a single source input frame (top) and user edit (bottom) we transfer the edit to frames of an animation where the camera moves along a path. (b) For moderate camera moves the edits transfer successfully. For severe camera movements (c) forward or (d) backwards the method reaches its limits as the scene geometry scales and/or new geometry comes into view. Note the artefacts in (c) where bricks in the archway are incorrectly turned turquoise or (d) only part of the table is brightened. For additional frames in the camera path please see supplementary material.

as the L_2 distance between their respective statistics vectors. We define the vector of differences between ROI $P_\star$ and neighbor region P_j across all render channels as $\mathbf{d}_j = \left(d_{j,1}, \ldots, d_{j,N}\right)^T$.

Our goal is to find a selection vector $\mathbf{x} = (x_1, \ldots, x_N)^T$ that makes the user-selected ROI unique with respect to its neighbor regions, i.e., maximizes the accumulated differences across all neighbor regions P_j:

$$\max_{\mathbf{x}} \sum_j \left(\mathbf{x}^T \mathbf{d}_j\right)^2 \qquad \text{s.t.} \quad \mathbf{x}^T\mathbf{x} = 1, \tag{7}$$

where $\mathbf{x}^T\mathbf{x} = 1$ is used to regularize the problem.

Such an optimal $\mathbf{x}$ can be directly computed as the eigenvector corresponding to the highest eigenvalue of the matrix $C = \sum_j \mathbf{d}_j \mathbf{d}_j^T$. Please refer to Appendix A for details. In order to convert the vector $\mathbf{x}$ to the final selection of channels X_A, we sort the channels based on x_i^2 and pick the top ones that accounts for $0.9 \sum_i x_i^2$. In our experiments, this resulted in typically $1-3$ selected channels.

5.1.3 Creating the edit mask. Using our selected channels X_A we use GrabCut [Rother et al. 2004] to create the final mask A'. Specifically, we composite the selected channels $\sum_{i \in X_A} A_i$ and sample pixels inside and outside the ROI $P_\star$ to form the mixture model for the foreground and background, respectively. We erode pixels from the ROI $P_\star$ boundary using a 3x3 kernel to avoid boundary artifacts during GrabCut, and discard pixels within 20% of the ROI $P_\star$ diameter. We then run GrabCut to get an edit mask (c.f., Figure 6 top-right). If desired, the user can adjust the edit mask using a brush tool. As a final step we set a mask pixel to zero if all of the selected channels are zero at that pixel. This prevents the adapted image analogies returning spurious correspondences when transferring the mask to the target view.

5.2 Parameterized Adjustments

To complete the edit e_A, our interface allows the user to adjust several parameters affecting the selected channels X_A in the region masked by A'. Additionally, after an edit has been transferred to the target scene (c.f., Sections 4.2 and 4.4), the user can similarly continue editing the transferred edit e_B. In Equation (6) we outline how the adjustment of a

parameter affects the final composite using the parameters θ. We currently support the following adjustments: exposure, levels, gamma, hue, saturation, lightness and Gaussian blur. These cover a wide range of edits as demonstrated by the variety of examples in our paper. Furthermore, comparing with the editing operations used in online tutorials [3DArtist 2016; CGalter 2015], the only editing operation we do not support is painting colors directly. The details of the specific parameters θ and how they are applied to a render channel can be found in Appendix C.

5.2.1 Grid Search. To find the optimal parameters θ in Equation (5), we first normalize the values of the individual parameters into the range $[0, 1]$ and perform a grid search sampling every 0.05, before denormalising into the original domain. In addition to sampling at every 0.05, we sample the exact parameter setting for the source view, as this may be the most appropriate parameter value for the transfer. We only perform grid search for parameters that are not at their default settings in the source view. We do not attempt to search for the Gaussian blur parameters and simply use the θ_A provided by the user in the source view edit.

6 RESULTS

In this section we show results of our automatic system for transferring parameterized edits. In our experiments, we used thirteen different 3D scenes, three of which were composited onto background photographs. Motivated by our target application, we selected 3D scenes that may appear in product images, such as a car model, a bottle, and a wristwatch. We created the majority of the scenes ourselves, using 3D assets we collected exclusively from Turbosquid[2] and Adobe Stock[3], with the exception of the *dragon*, made available by Stanford University[4]. Additionally, we used the San Miguel, Kitchen and Bathroom [McGuire 2011] scenes. When compositing a rendered view into a photograph, we used stock photographs as background images. The input renderings, user image-edits and source code can be found on the project webpage[5].

As there are no publicly-available datasets of 3D rendered scenes with 2D touchups, we manually set up and edited different 3D scenes to highlight a variety of touchups and effects that our automatic edit transfer approach can handle. Setting up the initial scene took between 30 and 120 minutes, with most of the time spent on adjusting lights and material properties.

We then applied common image-based edits to refine lighting effects, emphasize shape or material properties, and highlight important details and objects. Finally, to create the target views, we modified the 3D scenes in various ways, such as changing the camera viewpoint, re-arranging objects, and in some cases, replacing or adding object geometry. Please see Appendix D for a complete description of the edits to our 3D scenes.

Figure 7 shows the image-based edits and automatically transferred results for some of our example scenes. We show additional results in our supplemental PDF, which also includes the masks for all edits. These examples demonstrate the variety of different 2D touchups and scene modifications that our method is able to support. Since some of the edits are (by design) subtle, you may want to zoom into the electronic version of the paper and supplemental materials. Typically, finding edit-dependent weights (Section 4.3) takes 10 - 40 seconds to compute and the parameter grid search (Section 4.4) 20 - 60 seconds, the timings vary depending on the size of the mask.

[2]http://www.turbosquid.com
[3]https://stock.adobe.com
[4]https://graphics.stanford.edu/data/3Dscanrep/
[5]http://geometry.cs.ucl.ac.uk/projects/2017/edit-transfer/

6.1 Limitations

To test the limits of our method we transfer the edits from a single exemplar to frames from an animation sequence. In cases where features remain consistent throughout the animation, such as the rotating dragon in the supplemental video, the edits transfer successfully. However, if the content in the source and target views changes significantly throughout the sequence, the transfer begins to fail as the features in the source view are not present in the target view. We demonstrate this by zooming the camera in/out and revealing new geometry and lighting effects in the San Miguel scene (Figure 8).

In addition to the aforementioned limitation, we have identified four other potential limitations of our approach. First, the edit-dependent adaptive image analogies approach performs the synthesis in a coarse-to-fine fashion. As a result, features over small spatial extent may be missed by the coarse scales, resulting in mask synthesis artifacts, e.g., along an object boundary. Second, our approach may have difficulty in pixel regions when a second light source interferes with the target view. Thirdly, not all edit operations can be easily described using a mask and adjustment parameter (e.g. clone brush tool) and therefore cannot be transferred using our method. Finally, our formulation assumes the photometric render channels have a linear blending relationship, which may not be true for certain advanced edit operations.

6.2 Baseline Comparisons

For the edit transfer task, we compare our approach against a number of baselines and existing approaches for finding dense correspondences. Our first baseline is to simply use the known 3D shape correspondences between the two views (*correspondences*). The second baseline is StyLit [Fišer et al. 2016]. For StyLit, we compare against three variants: (i) "out of the box" StyLit that uses All Photometric render channels (*StyLit photo channels*), (ii) StyLit that uses only the edited render channel (*StyLit single*), and (iii) StyLit that uses all of the augmented render channels (*StyLit all channels*). We also compare against two algorithms for finding dense correspondences between two images using their source code: non-rigid dense correspondences (*NRDC*) [HaCohen et al. 2011] and Transfusive Image Manipulation (*Transfusive*) [Yücer et al. 2012].

We show output comparisons for NRDC, StyLit single, StyLit photo channels, and StyLit all channels in Figure 9. Notice how all baselines are unable to transfer the full edit from the source view to the target view for all cases. For example, all methods fail to remove the wine glass reflection in the background. NRDC and StyLit all channels introduce artifacts within the watch face. While all the baselines can transfer the car reflection, there are either artifacts in the transfer for the front light, or in the case of StyLit all channels the edit for the light fails to transfer at all.

Comparisons with Transfusive Image Manipulation are shown in Figure 10. The method was initialised with manually annotated pairs of points in the two views due to poor feature matching. Despite this additional interaction, the results suffer from inaccuracies in the correspondences and erroneously transfer edits.

We show two qualitative comparisons with additional baselines. In Figure 11 we show how using StyLit with All Photometric render channels, additionally augmented with the log of each channel and the Object Mask also fails to transfer edits correctly. Secondly, we show the effectiveness of our approach for adjustment parameter transfer in Figure 5. We compare against a baseline where we simply copy over the adjustment parameter the user selected in the source view to the target view. Notice that simply copying the parameter to the target view results in a bright reflection of the car. Our inferred adjustment parameter for the target view allows the reflection to more closely match the edited source view.

6.2.1 Perceptual study. To quantitatively evaluate our approach, we performed a perceptual study comparing our results against several baseline edit-transfer techniques: NRDC, StyLit single, StyLit photo channels, and StyLit all channels. We used a two-alternative forced choice (2AFC) design that shows a raw (A) and edited (A') source view, a raw target view (B), and two candidate edits for the target view generated by two of the methods under evaluation. The judge is asked to select the candidate edit that is more similar to A'. We generated all pairs of comparisons for three different scenes and ran the experiment on Amazon Mechanical Turk (AMT). In total, we had 147 distinct AMT workers and obtained 50 judgements for each pair of candidate edits. To analyze the data, we used the Bradley-Terry model [1952] to compute the likelihood of an edit transfer technique being selected by an AMT worker in a comparison. Our results are shown in Figure 12. Please refer to the supplemental for the interface shown to the AMT workers.

6.3 User Study

While the comparisons above demonstrate the effectiveness of our automatic edit-transfer technique, we also wanted to investigate the utility of our method within an interactive editing workflow where users may want to refine the automatic-transfer results. To this end, we conducted a comparative user study where participants used Adobe Photoshop to transfer edits to target scenes in two different ways: manually (i.e., specifying all the masks and image adjustment parameters from scratch) and using our automatic transfer results as a starting point. We use Photoshop in both conditions to achieve a more controlled comparison and provide an ecologically valid setting where users have access to an industry-standard set of editing features to refine auto-transferred edits. We recruited 16 participants from a university and a large software company for the study. Since our approach is designed primarily for artists with some image editing expertise, we focused on candidates who are reasonably familiar with Adobe Photoshop; ten of the participants had at least five years of Photoshop experience, and only two had used the software for less than a year. We report qualitative feedback from the editing sessions and quantitative data on the quality and completion time of the edits.

6.3.1 Methodology. We asked each participant to perform a total of four edit-transfer tasks on two different scenes, *Juice Bottle* (Figure 7) and *Car* (source view in Figure 7 and target view in Figure 9) . For each scene, we first presented a *source* Photoshop document containing both a raw (A) and edited (A') version of the scene, along with a text description of the edits with annotated figures highlighting the changes. In the Photoshop document, edits were represented as *adjustment layers* that encode a parameterized image adjustment and mask applied to a specified render channel. By toggling the visibility of these layers and the associated render channels, users were able to see the effect of each edit. They could also inspect the image adjustment parameters and masks.

After users familiarized themselves with the edits, we gave them a *target* Photoshop document with a modified configuration of the scene (B) and asked them to produce an edited version (B') that is analogous to the differences between A and A'. We created two types of target documents. The *manual* version provides the same set of adjustment layers (applied to the same set of render channels) as the source document, but each adjustment is set to its default parameters (which have no effect) and the mask is set to modify the entire image. This setup approximates current edit transfer workflows where users manually propagate each edit from source to target view by specifying the mask and image adjustment parameters from scratch. We also created an *automatic* version of the target document where the parameters and mask for each adjustment layer are initialized with the results of our automatic edit-transfer method. For each scene, we asked participants to transfer the edits using both the manual and automatic target documents to

produce a pair of edits $(B'_{\text{man}}, B'_{\text{auto}})$. We counterbalanced the order of the tasks to account for the potential learning effects from performing the edits twice.

We instructed users to complete the tasks as quickly as possible and recorded their completion times. To limit the duration of each session, we capped each task at ten minutes and alerted participants when they started to run out of time. After each task, we asked users to rate how well their B' matched A' as well as the perceived difficulty of the task on a 5-point scale. At the end of the session, we also asked whether they preferred the manual or automatic condition. Finally, in addition to these self-assessments, we obtained external judgements on the relative quality of each pair of user-generated edits $(B'_{\text{man}}, B'_{\text{auto}})$ using the same 2AFC design as the perceptual study described above.

6.3.2 Qualitative Feedback. Overall, participants expressed a clear preference for the automatic condition over the purely manual workflow. Amongst the 16 users, 14 indicated that they preferred the automatic version. They noted that working from the automatically transferred edits saved time and effort, even when they had to refine the masks and adjustment parameters. Our observations of the editing sessions support these sentiments; in the automatic condition, users spent far less time creating masks compared to the manual condition. The two participants who preferred the manual condition complained that they found it difficult to understand how some of the automatically-generated edits worked. However, both noted that they would probably prefer the automatic condition if they had created the original edits in the source view (which would typically be the case in real-world scenarios).

The self-assessments on the quality of edits and the difficulty of the tasks also clearly favour the automatic condition. For the Juice scene, only one of the 16 participants felt that the manual condition produced a better result than the automatic condition, and only three participants found the manual task easier than the automatic version. For the Car scene, two participants felt that their manual result was better, and one found the manual task easier.

6.3.3 Quality versus Completion Time. The task completion times and external quality judgments also support the qualitative findings. We visualize this data by encoding each $(B'_{\text{man}}, B'_{\text{auto}})$ pair generated by a given participant as a single (x, y) data point where x represents the difference in completion times and y represents the difference in the number of votes from the 2AFC comparison between the two conditions. In particular, $x = T_{B'_{\text{man}}} - T_{B'_{\text{auto}}}$, where T is completion time, and $y = V_{B'_{\text{auto}}} - V_{B'_{\text{man}}}$, where V is the number of votes. Using this encoding, Figure 13 provides a rates quality versus completion times for the two scenes.

The fact that most points lie in the top right quadrant indicates that users were generally faster and produced higher quality edits when starting with our automatically transferred edits. However, there are some differences in the relative quality of the manual and automatic results across the two scenes. For the Juice scene, all the automatic results received more votes, but for the Car scene, the votes are more evenly distributed. We believe the reason for this discrepancy is that the masks for the Car edits were easier to specify manually than the masks for the Juice edits, some of which required more careful brushing. Still, it is important to note that the automatic Car edits were at least comparable in quality to the manual edits, and participants consistently completed the edits much more quickly in the automatic condition, which is a key benefit of our approach.

7 CONCLUSION

We developed an interactive editing tool for 2D and 3D editing of rendered 3D scenes, which allows transfer of parametric 2D edits to new views of the scene or scenes with different objects. At the heart of our method is a new edit-dependent adaptive image analogies method. We demonstrated that our edit-dependent approach successfully transfers edits for a

variety of 3D scenes and 2D touchups, and outperforms prior approaches that rely on dense correspondences that do not take into account the user edits. Additionally, we evaluated the usefulness of our transfer method in a user study. Our tool opens up the possibility of additional functionalities that blur the boundary between 2D and 3D for editing, such as propagating 2D and 3D edits to automatically inferred 3D scene properties from the background photograph, e.g., to transfer edits to object shadows that affect others depicted in the background.

ACKNOWLEDGEMENTS

We thank our reviewers for their invaluable comments and the user study participants for their time and feedback. We also thank Moos Hueting, Tom Kelly, Carlo Innamorati, Aron Monszpart, Robin Roussel and Tuanfeng Yang Wang for their suggestions and support. We particularly thank Paul Guerrero for his help with the perceptual user study and Kaan Yücer for generating the Transfusive Image Manipulation [Yücer et al. 2012] comparisons. This work was partially funded by the ERC Starting Grant SmartGeometry (StG-2013-335373), EPSRC EngD Centre EP/G037159/1, a Rabin Ezra Trust Scholarship and gifts from Adobe.

A DERIVATION OF OPTIMAL CHANNEL SELECTION

As formulated in Section 5.1, our goal of determining a channel selection vector $\mathbf{x} = (x_1, \ldots, x_N)^T$ for a user-selected ROI can be cast as:

$$\max_{\mathbf{x}} \sum_j \left(\mathbf{x}^T \mathbf{d}_j\right)^2 \qquad \text{s.t.} \quad \mathbf{x}^T\mathbf{x} = 1.$$

Using Lagrangian multiplier λ, we can reformulate the above as $\max_{\mathbf{x}} E(\mathbf{x})$ where,

$$E(\mathbf{x}) := \sum_j \left(\mathbf{x}^T \mathbf{d}_j\right)^2 + \lambda(1 - \mathbf{x}^T\mathbf{x}).$$

Simplifying $E(\mathbf{x})$, we get:

$$\begin{aligned} E(\mathbf{x}) &= \mathbf{x}^T(\sum_j \mathbf{d}_j\mathbf{d}_j^T)\mathbf{x} + \lambda(1 - \mathbf{x}^\intercal\mathbf{x}) \\ &= \mathbf{x}^T\mathbf{C}\mathbf{x} + \lambda(1 - \mathbf{x}^T\mathbf{x}) \end{aligned}$$

with $C = \sum_j \mathbf{d}_j\mathbf{d}_j^T$. In order to find extrema of $E(\mathbf{x})$, we set

$$\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}} = 2\mathbf{C}\mathbf{x} - 2\lambda\mathbf{x} = 0.$$

Thus, to find an extrema of $E(\mathbf{x})$ we have to select an eigenvector of C. Let $\mathbf{x}_e$ be such an eigenvector, i.e., $C\mathbf{x}_e = \lambda\mathbf{x}_e$. For such a choice, $E(\mathbf{x})$ evaluates to

$$E(\mathbf{x}_e) = \mathbf{x}_e^T(C\mathbf{x}_e) = \mathbf{x}_e^T(\lambda\mathbf{x}_e) = \lambda,$$

where we used $\mathbf{x}_e^T\mathbf{x}_e = 1$ since $\mathbf{x}_e$ is an eigenvector.

Thus, to maximize $E(\mathbf{x})$, we have to pick the eigenvector with the *largest eigenvalue* among the N eigenvectors of C.

B RENDERER SPECIFIC AUGMENTED RENDER CHANNELS

The rendered of a scene using VRay took, on average, 10 minutes and Mitsuba on average took 8 hours. The list of Augmented Render Channels used for these examples can be seen in Table 1. Note that Mitsuba cannot separate lighting effects per light source but has a diverse set of non-photometric channels enabling our transfer method to work.

Table 1. Augmented Render Channels: The photometric render channels 1-3 are optionally rendered per light source. Geometry channels 4 -7 are rendered per light source. VRay object channels inner- and outer-distance transforms (DT) are generated in 2D using the object masks. We further augment the photometric render channels by adding the log of each channel. The Mitsuba outgoing- and incoming ray channels are the average rays for each pixel. We use All Photometric to refer to all channels in the Photometric column for a given renderer.

	Photometric	**Geometry**	**Object**
		VRay	
1.	Specular	Z-Depth	Object Mask
2.	Diffuse	Normals	Inner-DT
3.	Reflection	View Direction	Outer-DT
4.	Refraction	Normal View Cos	
5.	Global Illum	Light Direction	
6.	Shadow	Normal Light Cos	
7.	Caustics	Half Angle	
8.	Self Illum		
9.	Caustics		
10.	Subsurf Scatter		
		Mitsuba	
11.	Direct Specular	Z-Depth	Object Mask
12.	Direct Diffuse	Normals	Primitive Id
13.	Global Illum Specular	Outgoing Ray	BSDF Type
14.	Global Illum Diffuse	Incoming Ray	Sample Type
15.	Subsurf Scatter	OR Normal Cos	Albedo
16.	Environment Map	IR Normal Cos	
17.	Emitter	Half Angle	
18.	Caustics	Curvature	
19.		Raw Direct Illum	

C ADJUSTMENT PARAMETERS IMPLEMENTATION DETAILS

Exposure has a single scalar value $\theta \in [-10, 10]$ and adjusts the input as $f_\theta(B_i, p) = B_i(p) * 2^\theta$.

Levels require several parameters $\theta = \left(\theta_{in}^{min}, \theta_{in}^{max}, \theta_{out}^{min}, \theta_{out}^{max}, \theta^\gamma\right) \in [0, 1]^5$ and applied to a pixel as

$$f_\theta(B_i, p) = \left(\frac{B_i(p) - \theta_{in}^{min}}{\theta_{in}^{max} - \theta_{in}^{min}}\right)^{\frac{1}{\theta^\gamma}} \left(\theta_{out}^{max} - \theta_{out}^{min}\right) + \theta_{out}^{min}, \tag{8}$$

where γ allows for a single scalar parameter in the range $\theta \in [0, 10]$ and adjusts a pixel as $f_\theta(B_i, p) = B_i(p)^\theta$.
Hue, Saturation and Lightness changes are adjusted using the parameters $\theta = (\theta_h, \theta_s, \theta_l) \in ([-180, 180], [-180, 180], [-100, 100])$ and is applied to a pixel in HSL domain as $f_\theta(B_i, p) = B_i(p) + \theta$.

Gaussian blur is parameterized using a kernel size and standard deviation, $\theta = (\theta_x, \theta_\sigma)$ which is applied to a pixel by,

$$f_\theta(B_i, p) = \sum_{k=-\theta_x}^{\theta_x} \sum_{j=-\theta_x}^{\theta_x} G_{\theta_\sigma}(k, j) B_i(p') \tag{9}$$

where $p' = p + q$, $q = \begin{bmatrix} k & j \end{bmatrix}^\top$ and

$$G_{\theta_\sigma}(x, y) = \frac{1}{2\pi\theta_\sigma^2} e^{-\frac{x^2+y^2}{2\theta_\sigma^2}}. \tag{10}$$

D SCENE, EDIT AND TRANSFER DESCRIPTIONS

In this appendix we outline the scene rendering setup, the image-based edits performed, and the non-zero weighted render channels used to transfer edits to the target view.

D.1 Scenes

D.1.1 Car. The car scene was rendered with an area light source above the car and a point light source behind the camera. We made four image-based edits: (i) emphasize reflection on the ground; (ii) remove specular glare on the headlight; (iii) reduce specular reflection on the windscreen wipers; and (iv) reduce specular highlight on the bumper. We created many 3D scene variations: 3×viewpoint change, change in material, add duplicate geometry and new geometry. For each edit, the following channels were selected for transfer and reconstruction: (i) Log Specular (light source 1), Log Diffuse (light source 1), Log Specular (light source 2), Log Reflection; (ii) Specular (light source 1), Log Shadow, Normals, Normal View Cos; (iii) Specular (light source 2), Reflection, Global Illum; (iv) Specular (light source 1), Log Shadow, Log Reflection.

D.1.2 Juice Bottle. The bottle is lit by two point light sources, one behind the object and the other above and in front. We made three image-based edits: (i) remove specular glares around the outside of the bottle and push the specular highlight away from the bottle contour; (ii) remove harsh reflection on the bottom of the bottle; (iii) emphasize the label. In the target view, we changed the viewpoint. The following render channels were selected: (i) Log Specular, Outer-DT, Global Illumination; (ii) Reflection, Log Reflection; (iii) Log Reflection, Log Specular, Reflection, Self Illum.

D.1.3 Wine Bottle and Glass. The scene has 4 light sources: two area lights (one red and one white) and two point lights (one in front and one behind the wine and glass). We made five image-based edits: (i) remove big white reflection from the white area light reflecting off the table and wall; (ii) remove big red reflection from the red area light on the table; (iii) remove reflection of the wine glass on the back wall; (iv) emphasize bottom half of the white reflection on the wine bottle so it matches the reflection above; (v) remove distracting red refraction on the wine glass. In the target view, we changed the object geometry and viewpoint. The following render channels were selected: (i) Normals, Log Reflection (light 1), Reflection (light 1); (ii) Reflection (light 2), Normals; (iii) Reflection (light 1), Log Reflection (light 2), Log Reflection (light source 4), Log Refraction (light 2), Log Refraction (light 3); (iv) Refraction (light source 4), Log Reflection (light 4), Shadow, Log Global Illum; (v) Diffuse (light 2), Log Refraction (light 2), Log Reflection (light 2).

D.1.4 Watch. The scene is lit with a single point light source on the opposite side of the watch to the camera (not visible to camera). The image-based edits were (i) increase brightness of the watch face; (ii) add exaggerated highlight on the watch face; (iii) make metal material more reflective. In the target view, the watch was rotated and translated on the table. The selected render channels were (i) Log Reflection, Log Background; (ii) Log Refraction, Log Reflection, Log Diffuse, Log Shadow, Log Background; (iii) Log Global Illum, Log Shadow, Log Background, Log Refraction.

D.1.5 Whiskey. The original scene is rendered with a background photo of a beach with a directional light source above and to the left of the bottle. The image-based edits were (i) add halo effect around the outline of the bottle; (ii) emphasize the label to make it more visible. In the target view, we chose a different background image, the objects have been rotated and translated, and the light source position has moved to the right of the bottle. The following render channels were selected for the transfer: (i) Outer-DT, Normals, Log Background; (ii) Log Global Illum, Diffuse.

D.1.6 Dragon. There are three lights in this scene: two point lights on either side of the Dragon and a soft area light above. The image-based edits were (i) increase specular highlights to emphasize the Dragon's curvature; (ii) boost the GI channel inside the Dragon's body to give a glowing effect. In the target views, we rotated the Dragon 360°. The selected render channels were (i) Log Specular, Normals, Log Reflection, Diffuse; (ii) Log Global Illum, Inner-DT, Log Shadow.

D.1.7 Backpacks. The scene is lit from above by a single point light source in between the bags and the camera position. Three image-based edits were made: (i) make the fabric appear darker; (ii) remove unwanted highlight on the side of the bag; (iii) make creases of the bag orange matching the handle color. We modified the 3D scene by rotating and translating the two bags. The selected render channels were (i) Log Diffuse, Diffuse, GI; (ii) Log Specular, specular, Log Reflection, Log Shadow; (iii) Log Reflection, Normals, Log Shadow, Log Diffuse, Log Specular.

D.1.8 3D Text. The 3D text is composited into a background photo with a single point light behind the text. The material of the text is translucent. We edited the source view to emphasize translucency by increasing the exposure at some of the edges of the text. In the target view, the text has been rotated and translated, in addition to changing the background image. To transfer the edit, our method selects the Log Subsurf Scatter, Subsurf Scatter, Log Diffuse, Log Specular and Global Illum channels.

D.1.9 San Miguel. The scene's lighting comes from an environment map, which is only visible thought the atrium. The image-based edits were (i) increasing the exposure of the indirect global illumination channel to make the region in shade more visible (ii) adjusting the levels on the tree leaves to make them more prominent and green (iii) adjusting the hue, saturation and lightness of the wall. In the target views the camera moves and rotates revealing new geometry. To transfer the edits the selected channels were (i) Albedo, Sample Type, Log Global Illum Diffuse, Normals (ii) Albedo, Log Global Illum Diffuse and Primitive ID (iii) Albedo, Sample Type, and Normals.

D.1.10 Bathroom. The scene's lighting comes from an environment light outside, which is coming though the windows. The image-based edits were (i) reducing the exposure of the glass windows to make them appear frosted and (ii) hue, saturation and lightness of the floor to make it a different color. In the target view the camera is translated up and towards the left. To transfer the edits the selected channels were (i) BSDF Type, Raw Direct Illum and IR Normal Cos (ii) BSDF Type, Raw Direct Illum, Z-Depth and IR Normal Cos.

D.1.11 Kitchen. The scene's lighting comes from a sphere area light behind the camera position and several smaller point lights visible in the scene. The image-based edits were (i) blurring light sources, (ii) adjusting hue/saturation/lightness of the work surface, (iii) adjusting hue/saturation/lightness of the mug, (iv) reducing rendering noise on the pot using gamma correction and (v) reducing the specular highlights on the pot using the exposure parameter. In the target view the camera was translated. To transfer the edits the selected channels were (i) Albedo, curvature, Raw Direct Illum,

Sample Type (ii) Albedo, Normals, primitive id, Sample Type and Z-Depth (iii) Albedo, Primitive ID, Sample Type and Z-Depth (iv) Albedo, Raw Direct Illum, Z-Depth (v) BSDF Type, Raw Direct Illum, Z-Depth.

D.1.12 Skulls. The scene is lit by a large area light directly above the skulls. The image-based edits were (i) blurring the background skull to create a depth of field effect (ii) changing the hue, saturation and gamma to change the color and emphasize the reflections on the ground plane (Note this and edit (i) together is physically invalid, typical for our target application) (iii) adjusting the hue, saturation and lightness in the foreground skulls' eye sockets to make them appear to glow. In the target view the skulls were rotated and translated into a new configuration. To transfer the edits the following channels were selected (i) Log Diffuse, shadow, Log Shadow and Z-depth (ii) Log Diffuse, Log Reflection, Log Shadow (iii) Log Diffuse, Light Direction, Normal Light Cos, Half Angle

D.1.13 Instruments. This scene is lit by a white area light above the instruments and a red point light next to the camera. The image based edit (i) was blurring the background saxophone. In the target view the viewpoint was changed and the saxophone rotated. To transfer the edit (i) the Log Global Illum, Log Diffuse, Log Reflection, Inner-DT and Outer-DT were selected.

REFERENCES

3DArtist. 2016. Compositing 3D render passes in Photoshop. https://www.youtube.com/watch?v=wrsvFKNgxgU. (2016).

Xiaobo An, Xin Tong, Jonathan D. Denning, and Fabio Pellacini. 2011. AppWarp: Retargeting Measured Materials by Appearance-space Warping. *ACM SIGGRAPH Asia* 30, 6 (Dec. 2011), 10.

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM SIGGRAPH* 28, 3 (July 2009), 11.

Connelly Barnes and Fang-Lue Zhang. 2017. A survey of the state-of-the-art in patch-based synthesis. *Computational Visual Media* 3, 1 (2017), 17.

Connelly Barnes, Fang-Lue Zhang, Liming Lou, Xian Wu, and Shi-Min Hu. 2015. PatchTable: Efficient Patch Queries for Large Datasets and Applications. *ACM SIGGRAPH* 34, 4 (Aug. 2015), 10.

Gilbert Louis Bernstein and Wilmot Li. 2015. Lillicon: Using Transient Widgets to Create Scale Variations of Icons. *ACM SIGGRAPH* 34, 4 (July 2015), 11.

Floraine Berthouzoz, Wilmot Li, Mira Dontcheva, and Maneesh Agrawala. 2011. A Framework for Content-adaptive Photo Manipulation Macros: Application to Face, Landscape, and Global Manipulations. *ACM TOG* 30, 5 (Oct. 2011), 14.

Ralph Allan Bradley and Milton E. Terry. 1952. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika* 39, (3) (1952).

CGalter. 2015. Multipass Compositing in Photoshop - Vray Render Elements. https://www.youtube.com/watch?v=q0WpIzi8sc4. (2015).

Ming-Ming Cheng, Fang-Lue Zhang, Niloy J. Mitra, Xiaolei Huang, and Shi-Min Hu. 2010. RepFinder: Finding Approximately Repeated Scene Elements for Image Editing. *ACM SIGGRAPH* 29, 4, Article 83 (2010), 8 pages.

Ming-Ming Cheng, Shuai Zheng, Wen-Yan Lin, Vibhav Vineet, Paul Sturgess, Nigel Crook, Niloy J. Mitra, and Philip Torr. 2014. ImageSpirit: Verbal Guided Image Parsing. *ACM TOG* 34, 1, Article 3 (Dec. 2014), 11 pages.

Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. 2012. Image Melding: Combining Inconsistent Images using Patch-based Synthesis. *ACM SIGGRAPH* 31, 4 (July 2012), 10.

Olga Diamanti, Connelly Barnes, Sylvain Paris, Eli Shechtman, and Olga Sorkine-Hornung. 2015. Synthesis of Complex Image Appearance from Limited Exemplars. *ACM TOG* 34, 2 (March 2015), 14.

FastCompany. 2014. 75% Of Ikea's Catalog is Computer Generated Imagery. https://www.fastcodesign.com/3034975/75-of-ikeas-catalog-is-computer-generated-imagery. (2014).

Jakub Fišer, Ondřej Jamriška, Michal Lukáč, Eli Shechtman, Paul Asente, Jingwan Lu, and Daniel Sýkora. 2016. StyLit: Illumination-Guided Example-Based Stylization of 3D Renderings. *ACM SIGGRAPH* 35, 4 (July 2016), 11.

L. A. Gatys, A. S. Ecker, and M. Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. (June 2016), 9 pages.

Floraine Grabler, Maneesh Agrawala, Wilmot Li, Mira Dontcheva, and Takeo Igarashi. 2009. Generating Photo Manipulation Tutorials by Demonstration. *ACM SIGGRAPH* 28, 3 (July 2009), 9.

Paul Guerrero, Gilbert Bernstein, Wilmot Li, and Niloy J. Mitra. 2016. PATEX: Exploring Pattern Variations. *ACM SIGGRAPH* 35, 4 (July 2016), 13.

Paul Guerrero, Stefan Jeschke, Michael Wimmer, and Peter Wonka. 2014. Edit Propagation Using Geometric Relationship Functions. *ACM SIGGRAPH* 33, 2 (July 2014), 15.

Yoav HaCohen, Eli Shechtman, Dan B Goldman, and Dani Lischinski. 2011. Non-Rigid Dense Correspondence with Applications for Image Enhancement. *ACM SIGGRAPH* 30, 4 (July 2011), 9.

Yoav HaCohen, Eli Shechtman, Dan B Goldman, and Dani Lischinski. 2013. Optimizing Color Consistency in Photo Collections. *ACM SIGGRAPH* 32, 4 (July 2013), 9.

S.W. Hasinoff, M. Jozwiak, F. Durand, and W.T. Freeman. 2010. Search-and-replace editing for personal photo collections. In *IEEE International Conference on Computational Photography*. 8.

Paul S. Heckbert. 1990. Adaptive Radiosity Textures for Bidirectional Ray Tracing. *ACM SIGGRAPH* 24, 4 (Sept. 1990), 10.

James W. Hennessey and Niloy J. Mitra. 2015. An Image Degradation Model for Depth-augmented Image Editing. *SGP* 34, 5 (Aug. 2015), 9.

Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. 2001. Image Analogies. (Aug. 2001), 14 pages.

Shi-Min Hu, Kun Xu, Li-Qian Ma, Bin Liu, Bi-Ye Jiang, and Jue Wang. 2013a. Inverse Image Editing: Recovering a Semantic Editing History from a Before-and-after Image Pair. *ACM TOG* 32, 6 (Nov. 2013), 11.

Shi-Min Hu, Fang-Lue Zhang, Miao Wang, Ralph R. Martin, and Jue Wang. 2013b. PatchNet: A Patch-based Image Representation for Interactive Library-driven Image Editing. *ACM TOG* 32, 6 (Nov. 2013), 12.

L. Itti and C. Koch. 2001. Computational Modelling of Visual Attention. *Nature Reviews Neuroscience* 2, 3 (2001), 9.

Wenzel Jakob. 2010. Mitsuba renderer. (2010). http://www.mitsuba-renderer.org.

Anat Levin, Dani Lischinski, and Yair Weiss. 2004. Colorization Using Optimization. *ACM SIGGRAPH* 23, 3 (Aug. 2004), 6.

Ce Liu, Jenny Yuen, and Antonio Torralba. 2011. SIFT Flow: Dense Correspondence Across Scenes and Its Applications. *IEEE PAMI* 33, 5 (May 2011), 17.

Stephen Robert Marschner. 1998. *Inverse rendering for computer graphics*. Technical Report.

Morgan McGuire. 2011. Computer Graphics Archive. (August 2011). http://graphics.cs.williams.edu/data.

Fabio Pellacini, Frank Battaglia, Keith Morley, and Adam Finkelstein. 2007. Lighting with Paint. *ACM TOG* 26, 2 (June 2007), 10.

Ravi Ramamoorthi and Pat Hanrahan. 2001. A Signal-processing Framework for Inverse Rendering *(ACM SIGGRAPH)*. 12.

Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. 2001. Color Transfer between Images. *IEEE Comput. Graph. Appl.* 21, 5 (Sept. 2001), 8.

Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. "GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts. *ACM TOG* 23, 3 (Aug. 2004), 6.

Thorsten-Walther Schmidt, Fabio Pellacini, Derek Nowrouzezahrai, Wojciech Jarosz, and Carsten Dachsbacher. 2016. State of the Art in Artistic Editing of Appearance, Lighting and Material. *CGF* 35, 1 (2016), 17.

TheFoundry. 2017. Nuke. https://www.thefoundry.co.uk/products/nuke/. (2017). https://www.thefoundry.co.uk/products/nuke/

Yonatan Wexler, Eli Shechtman, and Michal Irani. 2007. Space-Time Completion of Video. *IEEE PAMI* 29, 3 (March 2007).

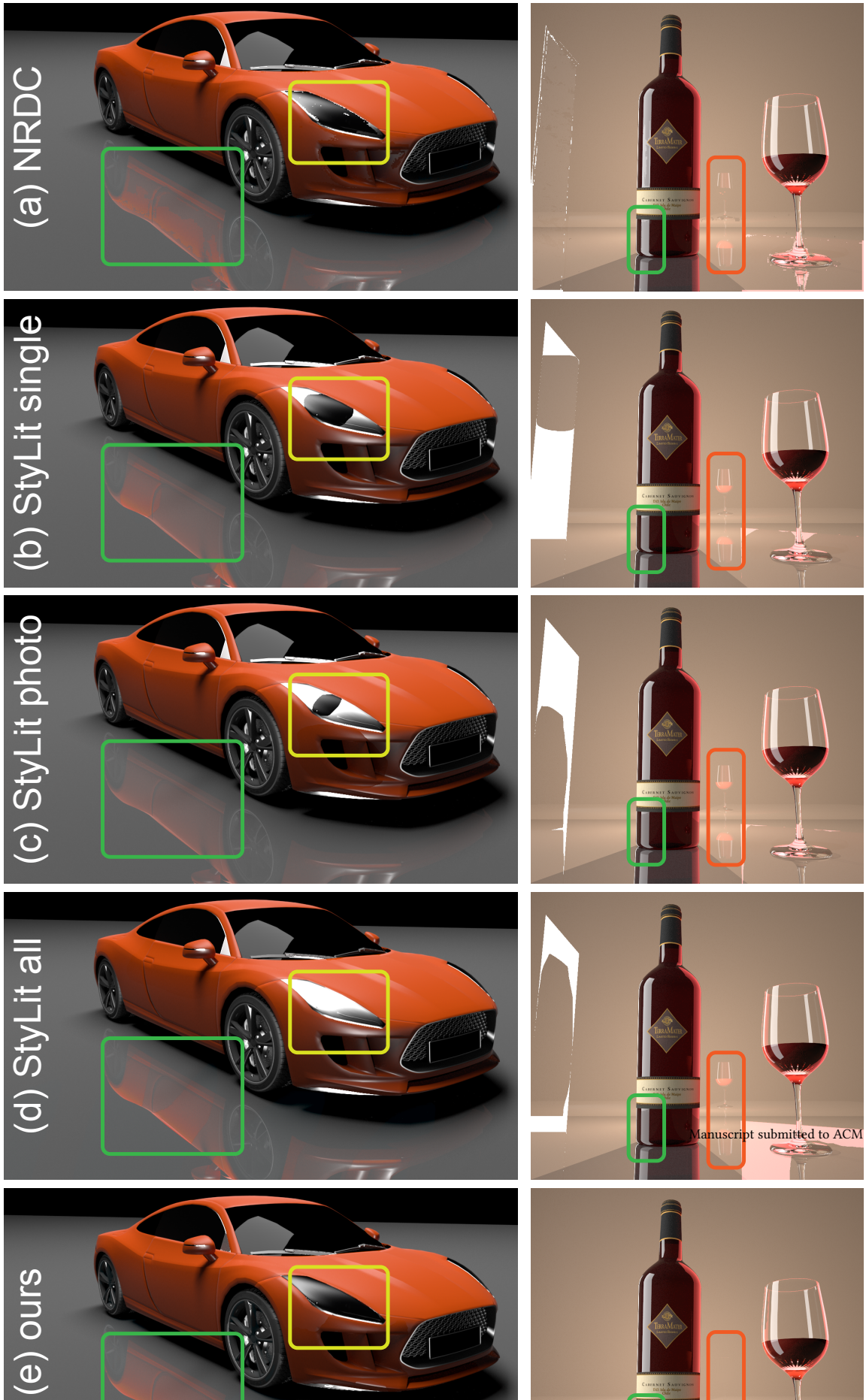
Jun Xing, Hsiang-Ting Chen, and Li-Yi Wei. 2014. Autocomplete Painting Repetitions. *ACM SIGGRAPH Asia* 33, 6 (Nov. 2014), 11.

Kaan Yücer, Alec Jacobson, Alexander Hornung, and Olga Sorkine. 2012. Transfusive Image Manipulation. *ACM SIGGRAPH Asia* 31, 6 (Nov. 2012), 11.

Kaan Yücer, Alexander Sorkine-Hornung, and Olga Sorkine-Hornung. 2013. Transfusive Weights for Content-Aware Image Manipulation. CGF Eurographics.

F. L. Zhang, J. Wang, E. Shechtman, Z. Y. Zhou, J. X. Shi, and S. M. Hu. 2016. PlenoPatch: Patch-based Plenoptic Image Manipulation. *IEEE TVCG* 23, 5 (2016), 14.

Youyi Zheng, Xiang Chen, Ming-Ming Cheng, Kun Zhou, Shi-Min Hu, and Niloy J. Mitra. 2012. Interactive Images: Cuboid Proxies for Smart Image Manipulation. *ACM SIGGRAPH* 31, 4 (2012), 11.


(a) NRDC
(b) StyLit single
(c) StyLit photo
(d) StyLit all
(e) ours

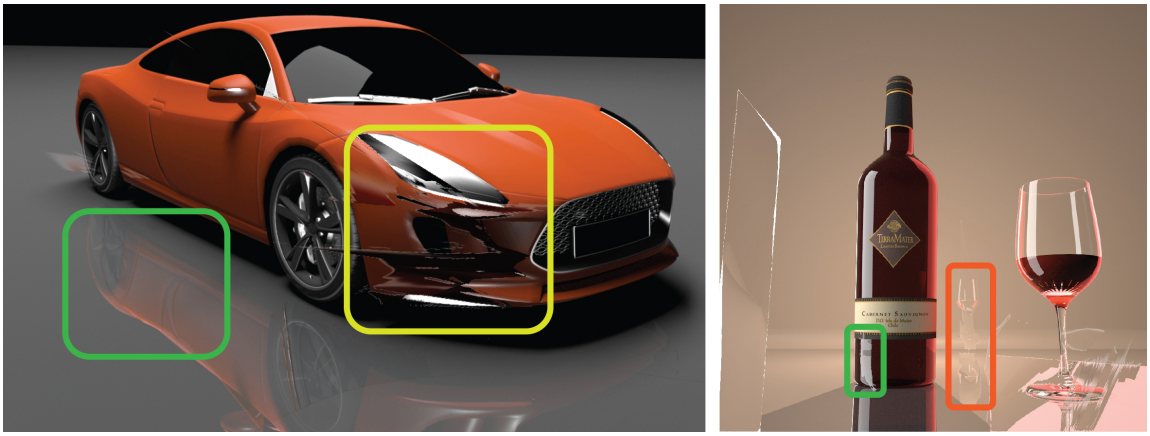


Fig. 10. Comparison with Transfusive Image Manipulation [Yücer et al. 2012]. In these examples, SURF matches (as used in their paper) failed to find reasonable correspondences and the method was initialized by manually selecting pairs of corresponding points between the source and target views. In the two examples the method fails to transfer all edits successfully and the edits which are transferred have ghosting artefacts . Please note that the tranfusive image manipulation work was designed for an entirely different application and it works directly on the composited image without access to the render channels.

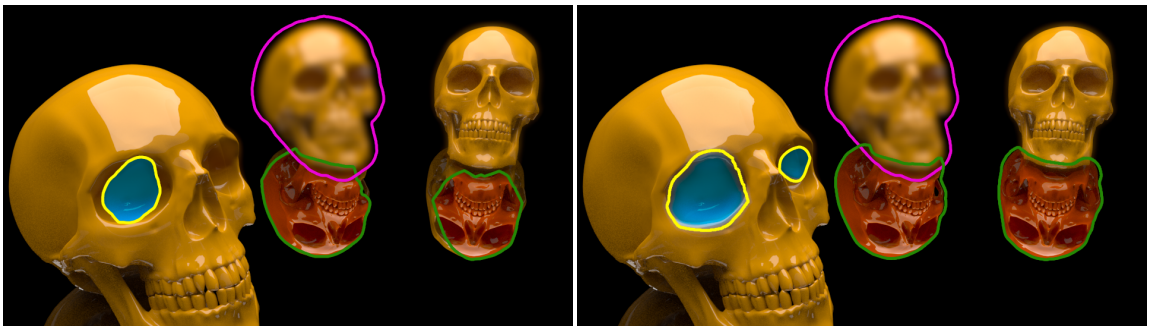


Fig. 11. Object Mask comparison. Using the source and target views from Figure 1 we show that (left) using StyLit with All Photometric render channels, additionally augmented with Object Mask fails to transfer the masks correctly. (right) Additionally adding the log of All Photometric render channels to the set to available channels improves results but still fails to transfer all edits correctly. Having Object Mask for guidance means edits can only be transferred to the same object they were applied to, in this example the desired outcome is to have both background skulls blurred as shown in our result in Figure 1.

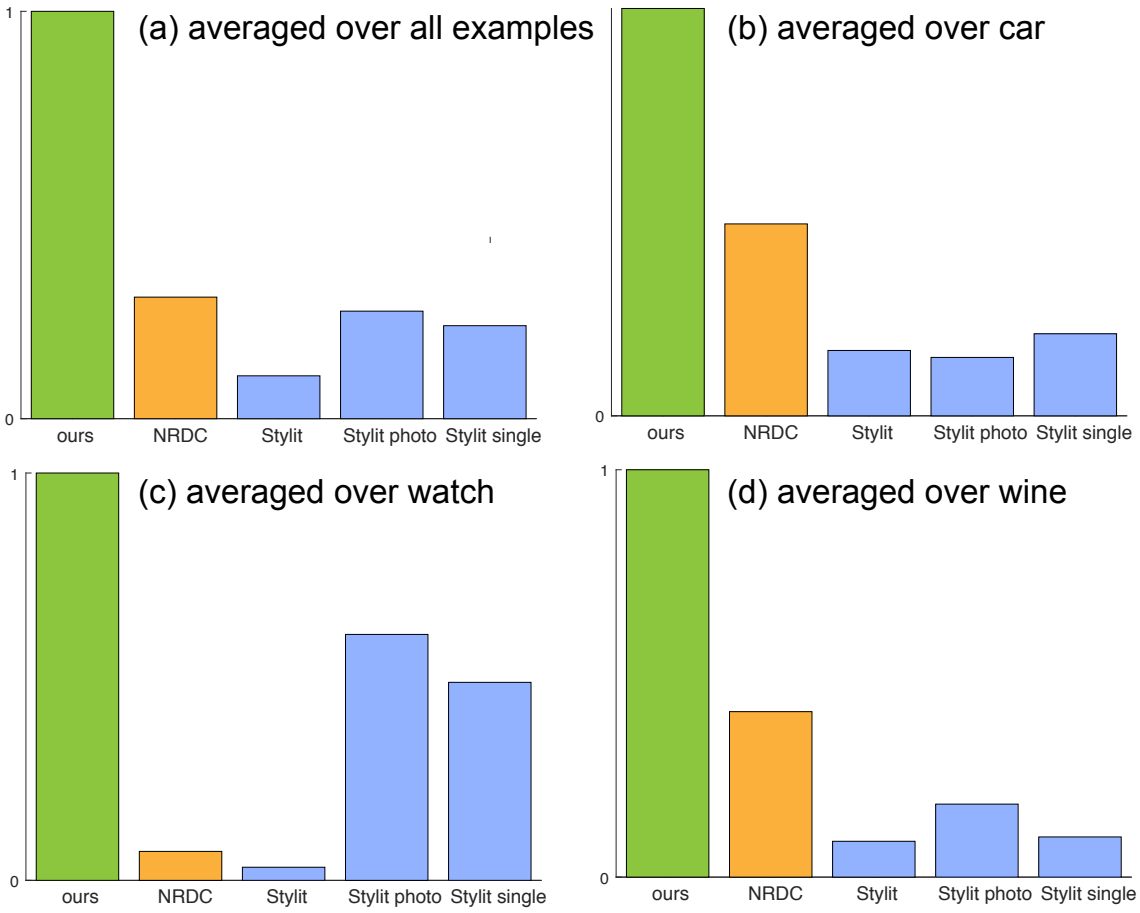


Fig. 12. Quantitative evaluation. We performed a pairwise-comparison user study on Amazon Mechanical Turk. Shown are likelihoods from the Bradley-Terry model [Bradley and Terry 1952] (normalized to 1) for the different approaches over (a) all scenes, (b) car scene, (c) watch scene, (d) wine scene. Please see the text for more details.

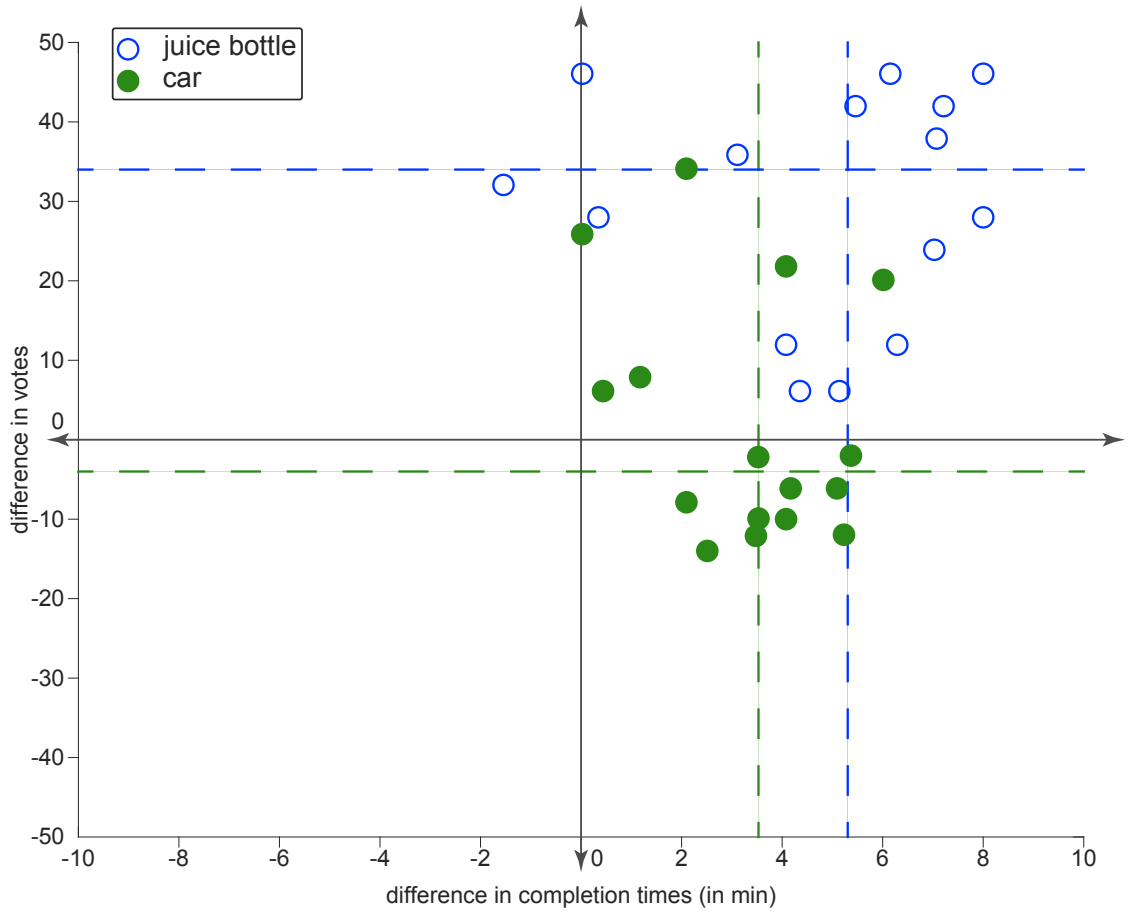


Fig. 13. Quality vs. Time: Scatterplot showing the difference in completion times and difference in number of votes. The dotted lines show the median value for each axis. The juice bottle (blue) results show significant improvement in completion time and quality. The car (green) has significant improvement in completion time but in quality the results are varied.